NBA Win and Playoff Predictions with Naive Bayes

September 28, 2019

0.1 Forecasting How Individual NBA Player Attributes Can Predict Whether Their Team Wins Enough to Make the Playoffs with Naive Bayes

In this project, I apply a Naive Bayes model to a NBA dataset from the Kaggle data repository (https://www.kaggle.com/noahgift/social-power-nba). The dataset is titled "Social Power NBA" and contains performance, salary, and twitter data for 100 NBA players of the 2016-2017 season.

To determine a cut-off number of wins that would be required to make the playoffs, I took a look at the FiveThirtyEight predictions for the NBA playoffs (link) and decided that at least 42 regular season wins are required to reach the playoffs for a given team. I chose this number also because it is a good measure of a winning team and is used as a benchmark of sucess in the NBA, since having greater than 41 wins would result in an overall winning record (>0.500 win percentage).

Below, I apply the Naive Bayes model to predict if a player will win enough to make the playoffs based on a number of attributes related to on and off court performance.

In [1]:
```
# Install necessary packages
import pandas as pd
import numpy as np

# Import train_test_split function
from sklearn.model_selection import train_test_split

# Import scikit-learn naive bayes model
from sklearn.naive_bayes import GaussianNB

#Import scikit-learn metrics module for accuracy calculation
from sklearn import metrics

# Import scikit-learn KFold module from the model selection package
from sklearn.model_selection import KFold

# Import scikit-learn modules necessary for k fold cross validation
from sklearn import svm
from sklearn.model_selection import cross_val_score
```

In [2]:
```
# Load NBA dataset
nba = pd.read_csv("~/Documents/UW Data Science Certificate/Methods for Data Analysis/M
```

```
# Take a look at the first five rows
nba.head()
```

Out[2]:

	PLAYER_ID	PLAYER_NAME	TEAM_ID	TEAM_ABBREVIATION	AGE	GP	W	\
0	201566	Russell Westbrook	1610612760	OKC	28	81	46	
1	1626246	Boban Marjanovic	1610612765	DET	28	35	16	
2	1627743	Demetrius Jackson	1610612738	BOS	22	5	1	
3	203076	Anthony Davis	1610612740	NOP	24	75	31	
4	201935	James Harden	1610612745	HOU	27	81	54	

	L	W_PCT	MIN	...	FGA_PG_RANK	FG_PCT_RANK	CFID	\
0	35	0.568	34.6	...	1	293	5	
1	19	0.457	8.4	...	356	47	5	
2	4	0.200	3.4	...	480	3	5	
3	44	0.413	36.1	...	3	95	5	
4	27	0.667	36.4	...	9	253	5	

	CFPARAMS	WIKIPEDIA_HANDLE	TWITTER_HANDLE	\
0	2,015,661,610,612,760	Russell_Westbrook	russwest44	
1	16,262,461,610,612,700	Boban_Marjanovi_	0	
2	16,277,431,610,612,700	Demetrius_Jackson	d_jay11	
3	2,030,761,610,612,740	Anthony_Davis_(basketball)	antdavis23	
4	2,019,351,610,612,740	James_Harden	jharden13	

	SALARY_MILLIONS	PTS	ACTIVE_TWITTER_LAST_YEAR	\
0	26.54	31.6	1	
1	7.00	5.5	0	
2	1.45	2.0	1	
3	22.12	28.0	1	
4	26.50	29.1	1	

	TWITTER_FOLLOWER_COUNT_MILLIONS
0	4.500
1	0.000
2	0.049
3	1.220
4	4.470

[5 rows x 63 columns]

1 Data Preparation for Naive Bayes

First, I one-hot encoded the Wins column based on whether or not the player had enough wins to make the playoffs (greater than or equal to 42). This one-hot encoded column will be used as the target for the remainder of the analysis since it's ultimately a yes or no question that we want to predict: Did the player win enough for their team to reach the playoffs?

Next, I removed features that were unique identifiers or were directly associated with the Wins

values. From this new dataframe, I designated all of the remaining columns as the features and kept the target assigned to the one-hot encoded playoffs column described above.

```
In [3]: # Generate a new target column with the wins one hot encoded for where greater than 42
        nba.loc[:, 'target'] = (nba.loc[:, 'W'] >= 42).astype(int)
```

```
In [4]: # Drop target column and features that are unique identifiers or directly related to n
        not_features = ['PLAYER_ID', 'PLAYER_NAME', 'TEAM_ID', 'TEAM_ABBREVIATION',
                        'WIKIPEDIA_HANDLE', 'TWITTER_HANDLE', 'CFPARAMS', 'W_RANK', 'L_RANK',
                        'W_PCT', 'W_PCT_RANK', 'W', 'target']

        # Create a dataframe with the dropped features removed
        nba_dropped = nba.drop(not_features, axis = 1)

        # Split data into features and targets
        features = nba.drop(not_features, axis = 1).values # features
        target = nba[["target"]].values # target
```

2 Naive Bayes

Using sklearn, I split the dataset into training and test subsets for both the features and the target attributes. I assigned 70% of the data as the training data and reserved 30% of the dataset for testing. I instantiated the Naive Bayes model and fit it to the features and target of the training subset. Using the testing features, I predicted the targets based on the Naive Bayes model and evaluated the accuracy of the model. Based on the individual player features, the model has a 80% accuracy in predicting whether the player's team will make the playoffs. This is a decent result and the accuracy may be misleading because there are many other factors as to why a team would make the playoffs outside of individual statistics. Things like team chemistry and intangible attributes like effort and heart are not quantified here.

```
In [5]: # Split dataset into training set and test set
        X_train, X_test, y_train, y_test = train_test_split(features, target,
                                           test_size=0.3,random_state=6) # 70% training and 30%
```

```
In [6]: # Instantiate model
        gnb = GaussianNB()

        # Train the model on the training sets only
        gnb_model = gnb.fit(X_train, y_train)
```

```
/Users/caseythayer/anaconda3/lib/python3.6/site-packages/sklearn/utils/validation.py:578: DataC
  y = column_or_1d(y, warn=True)
```

```
In [7]: #Predict the response for test dataset
        y_pred = gnb.predict(X_test)
```

```
In [8]: # Model Accuracy, how often is the classifier correct?
        print("Accuracy:",metrics.accuracy_score(y_test, y_pred))
```

```
Accuracy: 0.8
```

3 K Fold Cross Validation

I performed K Fold Cross Validation technique to confirm the accuracy of the Naive Bayes model. First, I defined the Kfold parameters by providing the number of folds to evaluate, this means that the dataset will be broken up into multiple subsets of the training data (containing the features and the target) and the accuracy of the model will be calculated for each subset to provide an overall accuracy of the model with a confidence interval.

I interrogated 10 folds and found that the overall accuracy of the Naive Bayes model prediction to be 86% with a 95% confidence interval of plus or minus 22%. This is a fairly large confidence interval and we can see that the accuracy that was determined above falls within the 95% confidence interval.

```
In [9]: # Define the number of folds and randomization for K Folds Cross Validation
        kfold = KFold(10, True, 1)
        # Perform K Folds Cross Validation and calculate cross validation scores
        clf = svm.SVC(kernel='linear', C=1)
        scores = cross_val_score(clf, X_train, y_train, cv = 10)
        print('Accuracy of each fold: ', scores)
        # Calculate the means of all cross validation scores to get overall accuracy and confi
        print('Accuracy : %0.2f (+/- %0.2f)' % (scores.mean(), scores.std() *2))
```

```
Accuracy of each fold:  [0.71428571 0.85714286 0.85714286 1.         1.         0.85714286
 0.71428571 1.         0.85714286 0.71428571]
Accuracy : 0.86 (+/- 0.22)
```

```
/Users/caseythayer/anaconda3/lib/python3.6/site-packages/sklearn/utils/validation.py:578: Data
  y = column_or_1d(y, warn=True)
/Users/caseythayer/anaconda3/lib/python3.6/site-packages/sklearn/utils/validation.py:578: Data
  y = column_or_1d(y, warn=True)
/Users/caseythayer/anaconda3/lib/python3.6/site-packages/sklearn/utils/validation.py:578: Data
  y = column_or_1d(y, warn=True)
/Users/caseythayer/anaconda3/lib/python3.6/site-packages/sklearn/utils/validation.py:578: Data
  y = column_or_1d(y, warn=True)
/Users/caseythayer/anaconda3/lib/python3.6/site-packages/sklearn/utils/validation.py:578: Data
  y = column_or_1d(y, warn=True)
/Users/caseythayer/anaconda3/lib/python3.6/site-packages/sklearn/utils/validation.py:578: Data
  y = column_or_1d(y, warn=True)
/Users/caseythayer/anaconda3/lib/python3.6/site-packages/sklearn/utils/validation.py:578: Data
  y = column_or_1d(y, warn=True)
/Users/caseythayer/anaconda3/lib/python3.6/site-packages/sklearn/utils/validation.py:578: Data
  y = column_or_1d(y, warn=True)
/Users/caseythayer/anaconda3/lib/python3.6/site-packages/sklearn/utils/validation.py:578: Data
  y = column_or_1d(y, warn=True)
/Users/caseythayer/anaconda3/lib/python3.6/site-packages/sklearn/utils/validation.py:578: Data
  y = column_or_1d(y, warn=True)
```

4 Conclusion

In this project, I employed the Naive Bayes model with the NBA dataset containing on and off court player attributes to predict whether or not a players team would make the playoffs.

I took the following steps to apply the Naive Bayes model to the NBA dataset to predict whether the on and off court individual player attributes would predict whether their team made the playoffs or not (greater than or equal to 42 wins). * One-hot encode wins column to represent whether the team achieved greater than or equal to 42 wins and passed the playoff threshold based on FiveThirtyEight NBA playoff predictions * Remove unique identifiers and features that were obviously associated with the number of wins * Split the dataset into training and test subsets based on assigned features and targets * Fit the Naive Bayes model to the training set and then use the features from the test dataset to predict the targets of the test dataset and evaluate the accuracy of these predictions * The model was 80% accurate in predicting whether the individual player attributes directly correlated with their team winning enough to make the playoffs * Further evaluate the accuracy of the model using K Fold Cross Validation in which 10 "folds" were obtained from the training subset (features and targets included), the model was fit and the accuracy was calculated for each fold. The average accuracy of all 10 folds and the 95% confidence interval was calculated to confirm the accuracy result determined above. * The K Folds Cross Validation results found that the average accuracy was 86% with a 95% confidence interval of plus or minus 22%. Our original accuracy result falls within this range so we can confirm that the accuracy of our model is correct based on the cross validation metrics.